UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

AN EXACT ALGORITHM FOR THE ASYMETRICAL CAPACITATED VEHICLE ROUTING PROBLEM

by

Gilbert Laporte*, Hélène Mercure*
and Yves Nobert**

* École des Hautes Études Commerciales de Montréal, 5255 avenue Decelles, Montréal, Canada H3T 1V6

** Département des Sciences administratives, Université du Québec à Montréal, 1495, rue Saint-Denis, Montréal, Canada H3C 3P8

The authors are grateful to the Canadian Natural Sciences and Engineering Research Council (grants A4747 and A5486) and to the Québec Government (F.C.A.C. grant 83EQ0428) for their financial support. Thanks are also due to Giorgio Carpaneto and Paolo Toth who made their TSP code [4] available to the authors.

Ce rapport a été publié grâce à une subvention du fonds F.C.A.C. pour l'aide et le soutien à la recherche.

Université de Montréal

Centre de recherche sur les transports - Publication #356
École des Hautes Études Commerciales - Cahier G-84-04

Février 1984

Dépôt légal - 3e trimestre 1984

ABSTRACT

The aim of this paper is to develop an exact algorithm for the asymmetrical capacitated vehicle routing problem, i.e. the multiple travelling salesman problem subject to capacity restrictions. The problem is solved by means of a branch and bound tree in which subproblems are modified assignment problems subject to some restrictions. Two branching rules and three partitioning rules are examined. Computational results for problems involving up to 260 nodes (cities) are reported.

UN ALGORITHME EXACT POUR LE PROBLEME DE TOURNEES AVEC CONTRAINTES DE CAPACITE

RESUME

Dans cet article, on présente un algorithme exact pour la résolution du problème de tournées avec contraintes de capacité. On résout le problème à l'aide d'un algorithme de "branch and bound" dans lequel les sous-problèmes sont des problèmes d'affectation modifiés assujettis à des contraintes de capacité. On compare deux critères de branchement et trois critères de partitionnement. On présente des résultats numériques pour des problèmes comportant jusqu'à 260 noeuds (villes).

1. INTRODUCTION

In the broadest sense, vehicle routing problems (VRP) consist of designing optimal routes for one or more vehicles through a set of locations (cities, nodes), subject to some restrictions. These problems underlie several distribution management situations and their study by operational research methods has led in recent years to major theoretical developments. Most VRP's are hierarchically related to the travelling salesman problem (TSP) which has received much attention in the last twenty-five years. Such an interest in the TSP lies partly in the sharp contrast between its deceivingly simple statement and its computational complexity. It can be stated as follows. Let $G = (N, A, C)$ be a graph where $N = \{1, \ldots, n\}$ is the set of nodes (cities), A the set of arcs (or edges) between the nodes and $C = (c_{ij})$, a matrix of distances defined over A. The TSP consists of determining the shortest route passing through each node once and only once. The TSP is NP-complete [11] and the largest problems consistently solvable by exact algorithms involve approximately 120 nodes [20] when C is symmetrical (i.e. $c_{ij} = c_{ji}$ for all $i,j \in N$) and 325 nodes otherwise [1]. Symmetrical problems are best handled by integer linear programming while asymmetrical problems can be solved most efficiently by network based methods which exploit the unimodular structure of the problem.

The aim of this paper is to describe an exact algorithm for one particular type of asymmetrical VRP. We first describe the problem.

Consider a graph G as above where C is asymmetrical and A, a set of *directed arcs*. To each node i, except the *depot* (city n, say), we associate a non-negative weight d_i ($d_n = 0$). The capacitated vehicle routing problem (CVRP) consists of establishing routes for m identical vehicles, each with capacity D and fixed cost f, in such a way that

(i) each vehicle starts and ends its journey at the depot;

(ii) each city (except the depot) is visited exactly once by one vehicle;

(iii) the sum of all weights on any given route never exceeds D;

(iv) the sum of fixed costs and routing costs is minimized.

CVRP's appear to be much more difficult to solve than TSP's involving the same number of cities. Bodin et al. [3] and Magnanti [18] describe various exact and approximate algorithms for the solution of several types of vehicle routing problems. It is noteworthy that efforts to develop optimal algorithms have been relatively limited and reflect in some sense the difficulty of the problem. Here is a brief summary of available exact algorithms for the CVRP.

Algorithms for the CVRP have been developed by Christofides [5] and by Christofides et al. [6,7]. In [5], Christofides considers a problem involving time and capacity constraints; he generates all feasible routes by means of a search tree; the largest problem solved to optimality involved 31 nodes. In [6], Christofides et al. solve the CVRP by means of a branch and bound procedure in which the bounds are either based on the computation of m-trees or of q-paths i.e. sequences of the form $(n, i_1, i_2, \ldots)$ where the i_k's are elements of $N-\{n\}$ and may not be all different and such that $\sum_k i_k \leq D$. The bounds derived in the latter case appear to be sharper [6]. The authors report exact solutions for problems of up to 25 cities. In a different paper, Christofides et al. [7] provide a branch and bound algorithm based on state-space relaxation, a variant of dynamic programming in which some of the states are relaxed. The bounds derived from this method appear to be quite strong: the lower bound at the root of the search tree was at 99.6% of the optimum in problems where $n = 25$. However, the branch and bound tree does not appear to have been developed fully and no optimal solution is provided. In a more recent paper [13], Laporte and Nobert suggest a branch and bound approach for symmetrical VRP's and report

exact solutions for problems involving up to 50 cities. Their algorithm is based on an integer linear programming formulation of the problem (similar to the one described in this paper) solved by constraint relaxation. Laporte et al. later use a similar approach to solve vehicle routing problems involving distance problems only [14] or capacity and distance constraints [15]. Finally, Fleischmann [9] also uses constraint relaxation to solve symmetrical CVRP's in which multiple visits to the same city are allowed and only a subset $\bar{N}$ of cities have to be visited. Fleischmann obtains optimal solutions for 4 problems out of 6, where $|\bar{N}|$ ranges from 28 to 68.

This paper describes a branch and bound method for the asymmetrical CVRP. It is based on an integer linear programming formulation of the problem (which incidentally can be derived by applying Benders decomposition [2] to the formulation proposed by Gavish and Graves [10] - see [18] and [19] for two different proofs). As in the work of Carpaneto and Toth [4], the problem is then solved by a branch and bound algorithm which exploits the underlying unimodular structure of the linear program.

2. FORMULATION

We first ignore the capacity constraints and consider the m-travelling salesman problem (m-TSP) [12] resulting from this relaxation. Provided an upper bound $\bar{m}$ for the value of m in the optimal solution is known, the m-TSP on G can be transformed into an equivalent TSP on an extended graph G' according to a simple set of rules. We use the transformation described by Lenstra and Rinnooy Kan [16].

(i) Increase the number of cities by introducing $\bar{m} - 1$ artificial depots; let $n' = n+\bar{m}-1$, $N' = \{1, \ldots, n'\}$, $N^* = (N'-N) \cup \{n\}$ and $A' = A \cup \{(i,j): i,j \in N', i \neq j, i \text{ or } j \in N'-N\}$.

(ii) Let $G' = (N', A', C')$ where $C' = (c'_{ij})$ is defined as follows:

$$(1)\quad c'_{ij} = \begin{cases} c_{ij} & (i,j \in N) \\ c_{in} & (i \in N,\ j \in N'-N) \\ c_{jn} & (i \in N'-N,\ j \in N) \\ \lambda & (i,j \in N^*) \end{cases}$$

The value of λ depends of the objective function considered for the problem:

$\lambda = \infty$ yields the minimum distance for $\bar{m}$ vehicles;

$\lambda = 0$ yields the minimum distance for at most $\bar{m}$ vehicles

$\lambda = -\infty$ yields the minimum distance for the minimum number of vehicles.

In the CVRP, the minimum number of vehicles required to visit the n cities is equal to

$$(2)\quad \underline{m} = \left\lceil \frac{\sum_{i=1}^{n-1} d_i}{D} \right\rceil$$

where $\lceil t \rceil = \begin{cases} \text{the smallest integer greater than or equal to } t \text{ if } t > 1; \\ 1 \text{ otherwise} \end{cases}$

Therefore, in problems where $\bar{m}$ is set equal to $\underline{m}$, the particular choice of λ is immaterial.

In what follows, let x_{ij} be a binary variable taking the value 1 if arc (i,j) is used in the optimal solution and the value 0 otherwise. Now consider the following four problems defined on G and on G'.

(P1) The modified assignment problem on G'

$$\text{minimize} \quad \sum_{i=1}^{n'} \sum_{j=1}^{n'} c'_{ij} x_{ij}$$

subject to

$$(3) \quad \sum_{\substack{i=1 \\ i \neq j}}^{n'} x_{ij} = 1 \qquad (j \in N')$$

$$(4) \quad \sum_{\substack{j=1 \\ i \neq j}}^{n'} x_{ij} = 1 \qquad (i \in N')$$

$$(5) \quad x_{ij} = 0 \text{ or } 1 \qquad (i,j \in N')$$

(P1) possesses a network structure and can be solved by an efficient algorithm (see for example the MAP procedure described in [4]).

(P2) The TSP on G'

(P2) is obtained by introducing the following subtour elimination constraints into (P1):

$$(6) \quad \sum_{\substack{i,j \in S \\ i \neq j}} x_{ij} \leq |S| - 1 \qquad (S \subset N',\ |S| \geq 2)$$

(P1) is a relaxation of (P2). In [4], Carpaneto and Toth solve (P2) by means of branch and bound algorithm based on this relaxation. We shall come back to this later.

(P3) The m-TSP on G

Optimal solutions to (P3) are obtained by first considering (P2) and applying to this problem the Carpaneto and Toth branch and bound algorithm until no illegal subtours remain. Subtours having a node set completely included in N* are legal and need not be eliminated. The solution to (P3) is then obtained by using the following rules:

(i) if $i \in N-\{n\}$ and $j \in N'-N$, replace (i,j) by (i,n);

(ii) if $i \in N'-N$ and $j \in N-\{n\}$, replace (i,j) by (n,j);

(iii) if $i,j \in N'-N$, remove (i,j).

(P4) The CVRP on G

This is the problem treated in this paper. Any feasible solution to (P4) is a solution to (P3) in which no subtour has a weight exceeding D. (P4) can be formulated by adding the following constraint to (P1):

$$(7) \qquad \sum_{i,j \in S} x_{ij} \leq |S| - V(S) \qquad (S \subset N - \{n\},\ |S| \geq 2)$$

where

$$(8) \qquad V(S) = \left\lceil \frac{\sum_{i \in S} d_i}{D} \right\rceil .$$

The validity of constraint (7) can be explained as follows. In any feasible solution to (P4), the number of vehicles required to visit all cities of a subset S of $N - \{n\}$ must be at least equal to V(S). The optimal solution will therefore contain at least V(S) inward arcs (from $\bar{S} = N'-S$ to S) and as many outward arcs (from S to $\bar{S}$). These conditions can be expressed as

$$(9) \qquad \sum_{i \in \bar{S},\, j \in S} x_{ij} \geq V(S) \qquad (S \subset N - \{n\},\ |S| \geq 2)$$

and

$$(10) \qquad \sum_{i \in S, j \in \bar{S}} x_{ij} \geq V(S) \qquad (S \subset N - \{n\},\ |S| \geq 2)$$

It is straightforward to show ([19], p.34) that when conditions (3) and (4) are satisfied, (9) and (10) are equivalent to (7). It will become clear in the next section (step 6) why in our formulation, we use (7) instead of (9) and (10).

We now describe an algorithm for (P4), using this formulation.

3. ALGORITHM

As in [4], we have chosen to solve the CVRP by first considering its relaxation provided by constraints (3) and (4) and by handling subtour elimination constraints (6) by means of a branch and bound tree, by successively fixing the x_{ij} variables at 0 or 1. The advantage of this procedure is obvious: as all subproblems are modified assignment problems, they possess a unimodular structure and therefore, an integer solution.

As our algorithm is derived from the work of Carpaneto and Toth, we have kept technical description to a minimum. We provide only a summary of our algorithm and, where necessary, describe the modifications made to the Carpaneto and Toth algorithm. Readers are referred to [4] for technical details. We first define the following notation.

z^* : the cost of the best feasible solution so far identified

z_h : the value of the objective function of the modified assignment problem at node h of the search tree

$\underline{z}_h$: a lower bound on z_h

I_h : the set of included arcs at node h of the search tree

E_h : the set of excluded arcs at node h of the search tree.

The algorithm can be summarized as follows.

Step 0: (initialization) Obtain a first value for z^* by means of an appropriate heuristic. We have used the Clarke and Wright algorithm [8]. Create node 1 of the search tree: set $I_1 = E_1 = \phi$, and obtain z_1 by solving the associated modified assignment problem. If $z_1 \geq z^*$, terminate. If the solution contains no illegal subtours, it constitutes the optimal tour: terminate. Otherwise, insert node 1 in the queue and go to step 1.

Step 1: (node selection) If the queue is empty, terminate. Otherwise, select the next node (node h) from the queue. In the *depth first* version of the algorithm, node h is selected according to a "last in first out" (LIFO) rule, thus reducing the memory requirements. In the *breadth first* version (used in [4]), branching is always on the pending node having the lowest z_h. This rule minimizes the number of subproblems solved. Go to step 2.

Step 2: (branching) The solution found at node h is illegal and must be eliminated by partitioning the current subproblem into descendant subproblems j characterized by sets I_j and E_j. Branching is made by breaking an illegal subtour according to a partitioning rule. The rules for the choice of this subtour and for its partitioning are described in section 4.

For each subproblem j, execute steps 3 to 6.

Step 3: (bounding) Compute a lower bound $\underline{z}_j$ on z_j according to step 3 of [4]. If $\underline{z}_j < z^*$, proceed to step 4. Otherwise, consider the next j and repeat step 3.

Step 4: (MAP solution) Solve the subproblem associated with node j (a modified assignment problem restricted by I_j and E_j). If $z_j \geq z^*$, consider the next j and proceed to step 3.

Step 5: (feasibility check) Check whether the current solution contains any illegal subtours: any subtour disconnected from N* is illegal; also, any chain between two depots having a total weight exceeding D corresponds to an illegal subtour in G. If the solution contains no illegal subtours, set $z^* = z_j$ and store the tour; if $z^* = z_h$, go to step 7; consider the next j and go to step 3. If the solution contains at least one illegal subtour, execute step 6.

Step 6: (check for inclusion of node j in the queue) The current solution contains subtours with nodes belonging to subsets S_k forming a partition of N'. For each k, it is valid to exclude up to $V(S_k)$ arcs from the subtour (see (7) and (8)). If at least one subtour is such that $V(S_k)$ exceeds the number of its arcs not already included in I_j, subproblem j has no feasible descendant. Thus, node j is inserted in the queue only if

$$(11) \quad V(S_k) \leq |S_k| - |S_k^2 \cap I_j| \text{ for all } k.$$

Step 7: (memory recovery: this step is only executed when estimated memory requirements for subsequent subproblems exceed the available memory) Eliminate from the queue all nodes h for which $z_h \geq z^*$. Proceed to step 1.

4. PARTITIONING RULES

Consider an illegal subtour identified in step 2 of the algorithm. (The rule used for the choice of this subtour is given later.) Suppose this subtour contains a set T of arcs not belonging to I_h:

$$(12) \quad T = \{(\alpha_1, \beta_1), \ldots, (\alpha_M, \beta_M)\}$$

It is assumed that these arcs are listed in the order in which they appear in the subtour where (α_1, β_1) is the first arc in lexicographic order. The rationale for this ordering will soon become clear. We studied three partitioning rules.

Rule 1 This rule is similar to the rule used by Carpaneto and Toth [4].

To the M immediate descendant nodes j from node h, are associated the following sets of included and excluded arcs.

$$(13)\quad I_j = \begin{cases} I_h & (j = 1) \\ I_h \cup \{(\alpha_u, \beta_u):\ u=1,\ldots,j-1\} & (j = 2, \ldots, M) \end{cases}$$

$$(14)\quad E_j^0 = E_h \cup \{(\alpha_j, \beta_j)\} \qquad (j = 1, \ldots, M)$$

When using this rule, branching is carried out from the subtour having the least value of M, i.e. the least number of immediate descendants. In the CVRP, it is valid to adjoin to E_j^0 the following sets $H(E_j^0)$ and $W(I_j)$ of excluded arcs.

(i) Replicas of excluded arcs between N* and N'-N*

$$(15)\quad H(E_j^0) = \bigcup_{k \in N^*} [\{(\alpha_i, k):\ (\alpha_i, \beta_i) \in E_j^0,\ \alpha_i \in N'-N^*,\ \beta_i \in N^*\} \cup \{(k, \beta_i):\ (\alpha_i, \beta_i) \in E_j^0,\ \alpha_i \in N^*,\ \beta_i \in N'-N^*\}]$$

(ii) Arcs excluded in order to prevent the occurence of subtours having a total weight exceeding D

In subproblem j, N' can be partitioned into $Q(N') = \{S_1, \ldots, S_\ell\}$ where $S_1, \ldots, S_\ell$ are subsets of nodes linked by the arcs of I_j. To

each subset S_t corresponds a chain having a weight

$$(16)\quad w(S_t) = \sum_{i \in S_t} d_i$$

and two end nodes p_t and q_t (which coïncide if $|S_t| = 1$). Let

$$(17)\quad e(S_t) = \{p_t, q_t\}.$$

Any two subsets of nodes S_t and S_v cannot belong to the same route in the optimal solution if their total weight exceeds D. It is therefore valid to exclude the arcs linking the end nodes of the chains associated with S_t and S_v. This is done by setting

$$(18)\quad W(I_j) = \{(i,j) \in [N-\{n\}]^2 :\ i \in e(S_t),\ j \in e(S_v),\ S_t, S_v \in Q(N'),\ w(S_t) + w(S_v) > D\}$$

$W(I_j)$ corresponds to the set of arcs forced to 0 in [13] and in [15]. It is worth noticing that the number of arcs included in $W(I_j)$ is related to the number of chains in I_j and to their weight. Since the arcs are successively included in the I_j's as they appear in the subtour, chains of arcs are fairly likely to be created in this manner.

As in step 5 of [17], E_j^0 is later augmented into E_j by the inclusion of all arcs which would certainly create illegal subtours if they were to appear in the solution.

Rule 2 The previous rule does not take into account the fact that, when an illegal subtour involving nodes of S has been identified, it can be eliminated by simultaneously excluding $r = \left\lceil \frac{\sum_{i \in S} d_i}{D} \right\rceil$ of its arcs in the descendant subproblems (see constraint (7)). Therefore, up

to $\binom{M}{r}$ immediate successors to node h can be created by taking all possible combinations of r arcs. (Note that r never exceeds M - see step 6 of the algorithm.) Consider the lexicographic enumeration of all combinations of arcs of T taken r at a time. Let T_j^r be the set consisting of the arcs in the j^{th} position of the list and $\bar{j}$, the index of the first of these arcs. Then

$$(19)\quad I_j = \begin{cases} I_h & (j = 1, \ldots, \binom{M-1}{r-1}) \\ I_h \cup \{(\alpha_1, \beta_1), \ldots, (\alpha_{\bar{j}-1}, \beta_{\bar{j}-1})\} & (j = \binom{M-1}{r-1}+1, \ldots, \binom{M}{r}) \end{cases}$$

and

$$(20)\quad E_j^0 = E_h \cup T_j^r \qquad (j = 1, \ldots, \binom{M}{r})$$

As above, E_j^0 is later augmented into E_j by $H(E_j^0)$, by $W(I_j)$ and according to step 5 of [17].

We tested two rules for the selection of the illegal subtour from which subproblems are created:

(i) as in rule 1, select the subtour for which M is minimum;

(ii) select the subtour for which $\binom{M}{r}$ is minimum.

The second rule appears to be the best.

Rule 3 Limited computational experience with rule 2 indicates that the number of nodes in the search tree can sometimes get out of hand, even if the bounds generated by this rule are in principle stronger than with rule 1. We have therefore considered a third partitioning strategy which combines the merits of the first two rules. First con-

sider a first set R of r arcs. Successively include each of its proper subsets and exclude the complement of that subset in R. Define the next I_j as $I_h \cup R$, the next E_j^0 as $E_h \cup \{(\alpha_{r+1}, \beta_{r+1})\}$ and the subsequent I_j's and E_j^0's as in rule 1. Formally, this rule is characterized by the following choices of I_j and E_j^0. Let

$$(21) \quad \sigma_\ell = \sum_{i=0}^{\ell} \binom{r}{i}$$

Then

$$(22) \quad I_j = \begin{cases} I_h & (j = 1) \\ I_h \cup T_{j-\sigma_\ell}^{\ell+1} & (\ell=0, \ldots, r-1;\ j=\sigma_\ell+1, \ldots, \sigma_{\ell+1})^* \\ I_h \cup T_1^r \cup \{(\alpha_u, \beta_u): u = r+1, \ldots, r+j-2^r\} & (j=2^r+1, \ldots, 2^r+M-r-1)^{**} \end{cases}$$

and

$$(23) \quad E_j^0 = \begin{cases} E_h \cup T_1^r - (I_j - I_h) & (j = 1, \ldots, 2^r-1) \\ E_h \cup \{(\alpha_{j-2^r+r+1}, \beta_{j-2^r+r+1})\} & (j=2^r, \ldots, 2^r+M-r-1)^{***} \end{cases}$$

As for the first two rules, E_j^0 is then transformed into E_j. Note that I_j contains relatively few arcs when $j \leq 2^r$. Therefore, in these cases, the cardinality of $W(I_j)$ is also likely to be small. But from this point of view, rule 3 is better than rule 2.

---

* If $r = M$, the largest value of j is 2^r-1.

** Undefined if $r \geq M-1$.

*** Undefined if $r = M$.

TABLE 1

Included and excluded arcs with three partitioning rules - an example

Rule	j	Indices of arcs in $I_j - I_h$	Indices or arcs in $E_j^0 - E_h$
1	1		1
	2	1	2
	3	1, 2	3
	4	1, 2, 3	4
	5	1, 2, 3, 4	5
2	1		1, 2, 3
	2		1, 2, 4
	3		1, 2, 5
	4		1, 3, 4
	5		1, 3, 5
	6		1, 4, 5
	7	1	2, 3, 4
	8	1	2, 3, 5
	9	1	2, 4, 5
	10	1, 2	3, 4, 5
3	1		1, 2, 3
	2	1	2, 3
	3	2	1, 3
	4	3	1, 2
	5	1, 2	3
	6	1, 3	2
	7	2, 3	1
	8	1, 2, 3	4
	9	1, 2, 3, 4	5

Table 1 shows the sets of included and excluded arcs for each of the three partitioning rules when M=5 and r=3. (I_h and E_h are not considered; arc (α_u, β_u) is represented by its index u.)

5. COMPUTATIONAL RESULTS

The algorithm presented in sections 3 and 4 was tested on several problems for which the weights d_i and the distances c_{ij} were generated from a uniform distribution on [0,100] and rounded up or down to the nearest integer. As in [13] and in [15], the vehicle capacity D was defined as

$$(24)\quad D = (1 - \alpha) \min_i \{d_i\} + \alpha \sum_{i=1}^{n} d_i$$

where α is a parameter *chosen* in [0,1]. Small values of α yield tight problems while large values produce looser problems. An algorithm such as ours, based on the relaxation of the capacity restrictions, is likely to perform better on the latter type of problems whereas approaches such as the one described in [6] which are based on the relaxation of degree constraints, are more effective in the former situation.

The value of m was determined as follows.

(i) In a first series of tests, m was *fixed* at its lowest feasible value

$$(26)\quad \underline{m} = V(N) = \left\lceil \frac{\sum_{i=1}^{n} d_i}{D} \right\rceil$$

i.e. n' was set equal to $n+\underline{m}-1$ and the value of λ in (1), to ∞.

(ii) Tests were also made by fixing m at $\underline{m}+2$ and $\underline{m}+4$ (λ was also set equal to ∞).

(iii) Finally, some problems were solved with an upper bound $\bar{m}$ on m. (We used $\bar{m} = \underline{m}+5$ and $\bar{m} = \underline{m}+10$). In these problems, n' was set equal to $n+\bar{m}-1$ and λ to 0.

In cases (i) and (ii), setting $\lambda = \infty$ causes no difficulty in practice since the algorithm systematically avoids all arcs having an infinite cost.

Six versions of the algorithm constructed by extending or otherwise modifying the Carpaneto and Toth program [4] were tested at various degrees for all combinations of the two node selection rules (breadth first and depth first) and of the three partitioning rules (excluding one arc at a time, r arcs at a time or the mixed version - see section 4). On the whole the depth first approach was far more inefficient than the breadth first approach (far too many subproblems had to be solved) and was soon dropped. The second partitioning rule (excluding r arcs at a time) is also much worse than the other two: it generates too many subproblems and in most of these, the number of included arcs is very small: this results in loosely constrained problems and poor bounds in the search tree. We have opted for the third partitioning rule which is slightly faster than the first one. For that rule, limited tests have shown that the best selection rule for the r arcs of E_j^0, (see (23) and table 1) is to choose the first r arcs not belonging to I_h, which appear on the subtour.

Tables 2, 3 and 4 contain computational results for problems solved by using the best version of the algorithm: breadth first node selection rule and third partitioning rule.

All problems were solved on the University of Montreal VAX 11/780 computer, using an FTN compiler. No time limit was imposed. The "working set" (i.e. the disc space available for central and virtual memory) was the only constraint.

In all tables, we report average results for successful problems. The meanings of the column headings are as follows.

SUCC : number of successful problems out of 5

TIME : CPU time in seconds

NODES: number of nodes of the search tree for which a lower bound $\underline{z}_h$ (see section 3, step 3) was computed

QUEUE: number of these nodes inserted in the queue (having a value of z_h less than z* at the time of their generation

DESC : number of nodes in the queue which were later examined, i.e. number of nodes having descendants

W : number of arcs belonging to the sets $W(I_j)$ (see (18)).

Table 2 presents results in the case where m is fixed at $\underline{m}$. For each n, and for a given problem, C remains the same for all values of α. The largest problems solved contained 90 nodes, which compares favorably with 50 cities in the symmetrical case [13] where a linear programming based algorithm was used. As expected, the difficulty of the problems decreased as α grew (when $\alpha = 1$ and when $m = \underline{m}$, the CVRP reduces to an m-TSP). Successful problems with $\alpha = 0$ were rare and are not reported in the tables; in the symmetrical case [13], the success rate for such problems was higher. This table also indicates that it would be quite efficient to solve large m-TSP's by means of the Carpaneto and Toth algorithm [4].

In table 3, we compare the behaviour of the algorithm for m fixed (at $\underline{m}$, $\underline{m}$+2 and $\underline{m}$+4) and for m bounded (by $\underline{m}$+5 and $\underline{m}$+10). For a given n and for a given problem, we have used the same distance matrix C in all cases. Problems with an upper bound on m appear to be the most difficult, followed by $m = \underline{m}$ and by $m = \underline{m}$+2 or $\underline{m}$+4. Similar behaviour was observed in [13]. This last case can easily be explained: when m is fixed at a value larger than $\underline{m}$, the total vehicle capacity far exceeds the sum of the weights d_i. The occurence of illegal subtours having a weight exceeding D is thus less likely in this case and the growth of the search tree more limited. This was confirmed by a third series of tests (table 4): m was fixed at $\underline{m}$+2 and $\underline{m}$+4 and problems solved for n = 100,

180 and 260. Computation times for $\alpha = .50$ and .75 are reasonably small but jump sharply when $\alpha = .25$.

This study has revealed that our approach is fairly efficient and can solve to optimality far larger problems than reported anywhere else in the operational research literature. We cannot however claim complete victory: some types of problems (namely those for which α is small and n is large) remain virtually intractable by exact algorithms.

TABLE 2

Problems for which $m = \underline{m}$

n	α	SUCC	TIME	NODES	QUEUE	DESC	W
10	0.25	5/5	93.671	1443.4	841.6	818.6	2290.6
	0.50	5/5	3.099	97.6	65.0	48.0	75.0
	0.75	5/5	3.124	109.2	58.4	51.8	11.2
	1.00	5/5	0.220	11.6	5.6	4.8	0
20	0.25	1/5	491.070	9648.0	6321.0	4012.0	57171.0
	0.50	5/5	33.836	635.4	327.6	205.0	253.2
	0.75	5/5	26.761	367.2	157.0	112.4	37.2
	1.00	5/5	0.375	20.4	8.2	5.8	0
30	0.50	4/5	83.920	661.5	304.2	155.7	114.0
	0.75	4/5	10.714	137.0	56.7	32.0	2.5
	1.00	5/5	1.950	31.0	18.8	12.2	0
40	0.50	4/5	904.099	5153.0	1967.0	925.7	1023.0
	0.75	5/5	552.093	3041.8	902.4	548.6	72.4
	1.00	5/5	1.643	24.8	13.8	7.0	0
50	0.50	4/5	458.677	5266.0	1179.0	908.5	510.5
	0.75	4/5	385.284	1530.2	302.7	216.0	21.0
	1.00	5/5	3.351	13.8	4.6	3.2	0
60	0.50	3/5	1175.576	3629.6	1363.6	502.6	1186.6
	0.75	5/5	1112.837	3342.2	1103.0	375.6	14.8
	1.00	5/5	5.542	41.6	23.0	9.2	0
70	0.50	2/5	109.349	138.0	101.5	16.0	189.0
	0.75	3/5	631.919	1364.3	383.3	144.0	29.3
	1.00	5/5	8.132	18.0	8.6	3.8	0
80	0.50	3/5	4718.441	7601.0	2353.3	886.6	1009.3
	0.75	3/5	2436.678	3569.6	598.0	462.3	16.6
	1.00	5/5	39.714	76.2	24.6	13.4	0
90	0.50	1/5	5787.598	7224.0	1186.0	680.0	898.0
	0.75	2/5	1162.316	3239.0	507.5	320.0	17.0
	1.00	5/5	16.020	22.2	10.2	5.4	0

TABLE 3

Comparison of the algorithm for various types of problems

	n	α	SUCC	TIME	NODES	QUEUE	DESC	W
EI = E	15	0.50	5/5	24.827	401.6	200.4	154.0	227.0
		0.75	5/5	10.216	281.8	118.4	99.8	43.4
	20	0.50	5/5	33.836	635.4	327.6	205.0	253.2
		0.75	5/5	26.761	367.2	157.0	112.4	37.2
2 + EI = E	15	0.50	5/5	0.154	6.8	2.0	1.6	0
		0.75	5/5	0.112	3.8	1.2	0.8	0
	20	0.50	5/5	0.226	5.6	2.4	1.6	0
		0.75	5/5	0.231	5.2	2.2	1.4	0
4 + EI = E	15	0.50	5/5	0.400	4.2	0.2	0.2	0
		0.75	5/5	0.168	1.2	0.2	0.2	0
	20	0.50	5/5	3.400	8.2	3.6	3.0	0
		0.75	5/5	2.039	8.2	3.4	3.0	0
5 + EI VI E	15	0.50	5/5	119.301	1035.6	428.2	354.4	372.2
		0.75	5/5	94.057	823.4	305.0	257.0	77.8
	20	0.50	5/5	110.290	689.2	309.8	192.4	207.6
		0.75	5/5	74.632	443.0	133.4	112.4	4.8
10 + EI VI E	15	0.50	5/5	126.864	694.2	282.2	243.4	266.6
		0.75	5/5	86.476	436.0	180.8	139.4	54.2
	20	0.50	5/5	365.830	2880.2	1062.8	921.2	534.2
		0.75	5/5	326.686	2312.2	755.4	687.8	123.4

TABLE 4

Problems for which $m = \underline{m} + 2$ and $\underline{m} + 4$

	n	α	SUCC	TIME	NODES	QUEUE	DESC	W
$m = \underline{m} + 2$	100	0.25	5/5	739.845	655.8	215.0	77.6	945.8
		0.50	5/5	98.285	143.2	22.0	14.0	0
		0.75	5/5	97.177	20.8	7.4	3.2	0
	180	0.25	4/5	372.299	1407.5	231.5	118.0	620.0
		0.50	5/5	57.319	212.8	24.4	13.4	9.2
		0.75	5/5	9.318	11.8	6.8	3.4	0
	260	0.25	5/5	4197.409	9729.2	751.8	595.0	1198.8
		0.50	5/5	89.889	43.8	20.2	4.0	0
		0.75	5/5	101.704	19.6	13.4	3.0	0
$m = \underline{m} + 4$	100	0.25	5/5	428.636	856.2	146.2	113.6	147.4
		0.50	5/5	8.152	5.4	3.6	1.2	0
		0.75	5/5	13.920	4.4	2.6	1.2	0
	180	0.25	4/5	81.303	676.7	100.7	68.5	80.7
		0.50	5/5	9.978	10.0	8.2	2.4	0
		0.75	5/5	5.441	6.0	4.2	1.8	0
	260	0.25	5/5	5119.091	7126.8	665.2	601.4	4077.4
		0.50	5/5	71.462	24.4	13.6	3.6	0
		0.75	5/5	65.125	20.4	12.2	3.4	0

REFERENCES

[1] BALAS, E., CHRISTOFIDES, N.: "A Restricted Lagrangean Approach to the Travelling Salesman Problem", Mathematical Programming, vol.21, pp.19-46, 1981.

[2] BENDERS, J.F.: "Partitioning Procedures for Solving Mixed-Variables Programming Problems", Numerische Mathematik, vol.4, pp.238-252, 1962.

[3] BODIN, L.D., GOLDEN, B.L., ASSAD, A., BALL, M.: "Routing and Scheduling of Vehicles and Crews. The State of the Art", Computers & Operations Research, vol.10, pp.69-211, 1983.

[4] CARPANETO, G., TOTH, P.: "Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem", Management Science, vol.26, no 7, pp.736-743, 1980.

[5] CHRISTOFIDES, N.: "The Vehicle Routing Problem", RAIRO (recherche opérationnelle), vol.10, pp.55-70, 1976.

[6] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations", Mathematical Programming, vol.20, pp.255-282, 1981.

[7] CHRISTOFIDES, N., MINGOZZI, A., TOTH, P.: "State-Space Relaxation Procedures for the Computation of Bounds for the Travelling Salesman Problem", Networks, vol.11, pp.145-164, 1981.

[8] CLARKE, G., WRIGHT, J.W.: "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", Operations Research, vol.12, pp.568-581, 1964.

[9] FLEISCHMANN, B.: "Linear Programming Approaches to Travelling Salesman and Vehicle Scheduling Problems", Paper presented at the XI. International Symposium on Mathematical Programming, 20 pp., Bonn, August 1982.

[10] GAVISH, B., GRAVES, S.C.: "The Travelling Salesman Problem and Related Problems", Working paper no.7905, Graduate School of Management, University of Rochester, 32 pp., 1979.

[11] KARP, R.M.: "Reducibility among Combinatorial Problems", Complexity of Computer Computations, pp.185-204, Plenum Press, 1972.

[12] LAPORTE, G., NOBERT, Y.: "A Cutting Planes Algorithm for the m-Salesmen Problem", Journal of the Operational Research Society, vol.31, pp.1017-1023, 1980.

[13] LAPORTE, G., NOBERT, Y.: "A Branch and Bound Algorithm for the Capacitated Vehicle Routing Problem", Operations Research Spektrum, vol.5, pp.77-85, 1983.

[14] LAPORTE, G., NOBERT, Y., DESROCHERS, M.: "Two Exact Algorithms for the Distance Constrained Vehicle Routing Problem", Networks, vol.14, pp.161-172, 1984.

[15] LAPORTE, G., NOBERT, Y., DESROCHERS, M.: "Optimal Routing under Capacity and Distance Restrictions", Cahiers du GERAD, G-83-18, Ecole des Hautes Etudes Commerciales de Montréal, 23 pp., 1983.

[16] LENSTRA, J.K., RINNOOY KAN, A.H.G.: "Some Simple Applications of the Travelling Salesman Problem", Operational Research Quarterly, vol.26, pp.717-734, 1975.

[17] LITTLE, D.C., MURTY, K.G., SWEENEY, D.W., KAREL, C.: "An Algorithm for the Travelling Salesman Problem", Operations Research, vol.11, pp.972-989, 1963.

[18] MAGNANTI, T.L.: "Combinatorial Optimization and Vehicle Fleet Planning: Perspectives and Prospects", Networks, vol.11, pp.179-214, 1981.

[19] NOBERT, Y.: "Construction d'algorithmes optimaux pour des extensions au problème du voyageur de commerce", Thèse de doctorat, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, 240 pp., 1982.

[20] PADBERG, M.W., HONG, S.: "On the Symmetric Travelling Salesman Problem: a Computational Study", Mathematical Programming Study, vol.12, pp.78-107, 1980.